Assignment 2 – Hangman Report Template

Arushi Tyagi

CSE 13S – Winter 24

Purpose

Audience for this section: Pretend that you are working in industry, and write this paragraph for your boss. You are answering the basic question, “What does this thing do?”. This section can be short. A single paragraph is okay.
The hangman game takes a secret phrase or word that one user inputs in. That secret is stored in the program and printed with empty underscores, denoting what the sentence is. For example, if the secret word is ”Hello” it would print _ _ _ _ _. However, if there are hyphens or apostrophe’s they will be displayed. Then another user inputs a guess for a letter. The program either says it’s wrong and adds it to the ”Eliminated” list or if the letter guess is correct, it’ll replace the empty dash with the letter corresponding to it. If the user incorrectly guesses a letter 6 times, then the user loses.

Questions

Please answer the following questions before you start coding. They will help guide you through the assignment. To make the grader’s life easier, please do not remove the questions, and simply put your answers below the text of each question. To fill in the answers and edit this file, you can upload the provided zip file to overleaf by pressing [New project] and then [Upload project].

Guesses

One of the most common questions in the past has been the best way to keep track of which letters have already been guessed. To help you out with this, here are some questions (that you must answer) that may lead you to the best solution.

- How many valid single character guesses are there? What are they?
- Do we need to keep track of how many times something is guessed? Do you need to keep track of the order in which the user makes guesses?
- What data type can we use to keep track of guesses? Keep in mind your answer to the previous questions. Make sure the data type you chose is easily printable in alphabetical order. [1]
- Based on your previous response, how can we check if a letter has already been guessed. [2]

A list called ”guessed” with all the previously guess letters can exist. There are 26 valid guesses (26 letters in the alphabet), so it would be easy to define the list given its max length. You don’t need to keep track of the order because after every guess, the program can check if the guess has already been made by iterating through the list. In order for it to be alphabetical, we can look at the ASCII values of all the guessed numbers and arrange them from smallest to largest.

[1]Your answer should not involve rearranging the old guesses when a new one is made.
[2]The answer to this should be 1-2 lines of code. Keep it simple. If you struggle to do this, investigate different solutions to the previous questions that make this one easier.

Strings and characters

- Python has the functions `chr()` and `ord()`. Describe what these functions do. If you are not already familiar with these functions, do some research into them.
  The chr() function takes an integer and returns the corresponding character. The ord() function does the opposite and returns the integer corresponding to the character given.

- Below is some python code. Finish the C code below so it has the same effect. [3]

  ```
  x = 'q'
  print(ord(x))
  ```

  C Code:

  ```
  char x = 'q';
  printf("%d\n", (int)x);
  ```

- Without using `ctype.h` or **any** numeric values, write C code for `is_uppercase_letter()`. It should return false if the parameter is not uppercase, and true if it is.

  ```
  #include <stdbool.h>
  char is_uppercase_letter(char x){
      if (x >= 'A' && x <= 'Z'){
          return true;
      }
      else{
          return false;
      }
  }
  ```

- What is a string in C? Based on that definition, give an example of something that you would assume is a string that C will not allow.
  A string in C is an array of characters. It ends with a null terminator '\0' A string that C will not allow is an array of characters without a null terminator.

- What does it mean for a string to be null terminated? Are strings null terminated by default?
  When a string is null terminated it means that it is the end of the string. If you write a string in double quotes it is null terminated by default but if you write a string as an array of separate characters, you need to add a null terminator because it does not do it by default.

- What happens when a program that is looking for a null terminator and does not find it.
  It will print an error because it cannot find the null terminator.

- In this assignment, you are given a macro called `CLEAR_SCREEN`. What is it's data type? How and when do you use it?
  It is a function that is used to clear the console screen. You just print

  ```
  CLEAR_SCREEN();
  ```

  This calls the function and implements it.

Testing

List what you will do to test your code. Make sure this is comprehensive. [4] Remember that you will not be getting a reference binary [5].

[3]Do not hard code any numeric values.

[4]This question is a whole lot more vague than it has been the last few assignments. Continue to answer it with the same level of detail and thought.

[5]The output of your binary is not the only thing you should be testing!

- If the secret word is more than 256 letters. It should print ”The secret phrase is over 256 characters.”
- If the secret word has no arguments or too many (without double quotes).It should print ”wrong number of arguments”
- If the character guess is uppercase (should be lowercase)
- If the character guess is a character that is not a letter

How to Use the Program

Audience: Write this section for the user of your program. You are answering the basic question, “How do I use this thing?”. Don’t copy the assignment exactly; explain this in your own words. This section will be longer for a more complicated program and shorter for a less complicated program. You should show how to compile and run your program. You should also describe any optional flags or inputs that your program uses, and what they do.
In the command line, the user needs to enter a secret phrase that is either one word or a sentence enclosed in double quotes. The phrase should be lowercase but can contain apostrophes and hyphens and it also needs to be 256 characters or less. Then, once the program is compiled and is running, another user will need to guess what the secret phrase is one letter at a time. They will be prompted to enter in a letter and the program will output if it is correct or not. If it is correct, the blank that holds the letter guessed will reveal the letter. This will continue until the user has correctly guessed the whole phrase or has incorrectly guessed six times. If the user did not get it, the program will print the secret phrase and a statement that says the user lost. If the user got it, then it will print a statement that says the user won!

Program Design

Audience: Write this section for someone who will maintain your program. In industry you maintain your own programs, and so your audience could be future you! List the main data structures and the main algorithms. You are answering the basic question, “How is this thing organized so that I can have a chance of fixing it?”. This section will be longer for a more complicated program and shorter for a less complicated program.
There are two main files: hangman_helpers.c and hangman.c. The first file (hangman_helpers.c) will have most of the functions that are required in this program. It will check whether the secret and letters are valid inputs. The functions will mainly use three arguments: s (secret), c (the letter input), and *s (the secret pointer). It’ll also have a function to check if there user has won or lost. Finally, it will have functions to keep track of the letters already guessed through a list. The hangman.c file will have the main function which iterates through the game. It’ll print the empty dashes for the secret phrase and check if the user has entered the right letter. If yes, then it’ll print the letter in the correct place and if it’s wrong, it’ll tell the user it’s wrong and print the Hangman art. It will also call all the functions from hangman_helpers.c to help with iterating through the game.

Pseudocode

Give the reader a top down description of your code! How will you break it down? What features will your code have? How will you implement each function.
In the file hangman_helpers.c, the function to check if the string contains the character guessed will iterate through the length of the secret string. It’ll check if the secret pointer equals the character guessed and return a boolean true or false. There will be another function to take in the letter input and another function to check if it is a valid lowercase letter and return a boolean. Similarly, there will be a boolean to validate the secret input and return a true or false boolean. The check winner function will check if all the characters in the secret string have been guessed by checking if there are still any _ characters left. In the main function in hangman.c the program will take in the secret input and have the arguments

```
int argc, char* argv[]
```

The program would print the secret phrase with its corresponding dashes and hyphens/apostrophes if needed. It will achieve this by iterating through the length of the string and using an if statement to print the non-letter characters. Then, it'll check if the letter guess input is contained in the string by calling the function to do so. If it is, then it'll replace it, but if not it'll add the letter to the Eliminated list. It'll also use an if statement to check if the guessed letter has already been guessed by checking it against a ”guessed” list. It would keep going until the user has lost or won and the corresponding print statement would be displayed.

Function Descriptions

For each function in your program, you will need to explain your thought process. This means doing the following

- The inputs of every function (even if it's not a parameter)
- The outputs of every function (even if it's not the return value)
- The purpose of each function, a brief description about a sentence long.
- For more complicated functions, include pseudocode that describes how the function works
- For more complicated functions, also include a description of your decision making process; why you chose to use any data structures or control flows that you did.

Do not simply use your code to describe this. This section should be readable to a person with little to no code knowledge. **DO NOT JUST PUT THE FUNCTION SIGNATURES HERE. MORE EXPLANATION IS REQUIRED.**

- Function: string_contains_character. Inputs: *s (string pointer), c (letter guess). Output: boolean (true or false). Purpose: checks if the string pointer equals the character. This checks if the guessed letter is in the secret phrase.
- Function: read_letter. Input: c (letter guess from user). Output: letter guessed. Purpose: returns the guessed letter into the main function.
- Function: is_lowercase_letter. Input: c (letter guessed). Outputs: boolean (true or false). Purpose: Checks if the letter input is lowercase and valid.
- Function: validate_secret. Input: *s (secret string pointer) Output: boolean (true or false). Purpose: Checks if the secret string is valid (lowercase letters, apostrophes, hyphens accepted).
- Function: check_winner. Inputs: secret. Output: boolean (true or false). Purpose: checks if there is a winner or not. If the secret has been guessed return true. Else return false.
- Function: main. Inputs: argc, argv (for secret), letter guess. Outputs: status after every guess, eliminated list, hangman image, win or lose,. Purpose: Iterates through the game and takes in the guess of the user. If correct, prints the letter in the correct space. If incorrect, adds the letter to eliminated list and prints hangman image. When eliminated ¿= 6 or secret is guessed, prints win or lose message.

References